Appl. No. 10/697,911
Response dated 09/01/2006
Reply to Office Action of 06/02/2006

IN THE CLAIMS:

This listing of claims will replace all prior versions, and listing of claims, in the Application.

Listing of claims:

1. (Currently amended) A computer-implemented method of editing data having a fixed ~~first~~ format on a computer system that uses data having a non-fixed format such that after being edited, the data continues to have the fixed format, the method comprising the steps of:

~~(a)~~ receiving, by the computer system, a ~~first~~ data byte array, the data byte array having the non-fixed format and including data for editing a fixed-length statement;

~~(b)~~ determining the fixed format encoding of the ~~data byte array~~ of the fixed-length statement by determining a number of bytes in each of a plurality of fixed-length fields that comprise ~~a~~ the fixed-length statement;

~~(c)~~ determining a number of bytes in the fixed-length statement;

~~(d)~~ creating a ~~first~~ data string from the ~~first~~ data byte array, the data string being encoded using the determined fixed format encoding, given a starting byte position and the number of bytes in the fixed-length statement; ~~and~~

~~(e)~~ assigning an attribute to each byte of the ~~first~~ data string; and

editing the fixed-length statement using the data string.

2. (Currently amended) The computer-implemented method of claim 1, further comprising the step of repairing an ending of the ~~first~~ data string.

3. (Currently amended) The computer-implemented method of claim 2, wherein the step of repairing the ~~end~~ ending of the ~~first~~ data string comprises the steps of:

~~(a)~~ determining ~~if~~ whether or not the last byte of the ~~first~~ data string is a second byte of a double byte character, and if the last byte of the data string is the second byte of a double byte character ~~so~~, setting a value of the second to the last byte to a shift-out character and changing the attribute of the ~~a~~ second to the last byte to ~~be~~ a shift-out character and removing the last byte;

~~(b)~~ ~~if not, then~~ determining, if the last byte of the data string is not a second byte of a double byte character, whether or not ~~if~~ the last byte of the ~~first~~ data string is a first byte of a double byte character and if the last byte of the data string is a first byte of a double byte character ~~so~~, then setting the value of the last byte to a shift-out character and changing the attribute of the last byte to ~~be~~ a shift-out character;

~~(c)~~ ~~if not,~~ determining, if the last byte of the data string is not a first byte of a double byte character, whether or not ~~if~~ the last byte of the ~~first~~ data string is a shift-out character, and if the last byte of the data string is a shift-out character ~~so~~, then removing the last byte of the ~~first~~ data string;

~~(d)~~ ~~If not~~ determining, if the last byte of the data string is not a shift-out character, whether or not ~~if~~ the last byte of the ~~first~~ data string is a shift-in character and if the last byte of the data string is a shift-in character ~~so~~, determining if ~~a~~ the second to the last byte is a shift-out character, and if the second to the last byte is a shift-out character ~~so~~, removing the last two bytes of the ~~first~~ data string.

4. (Currently amended) The computer-implemented method of claim 1, further comprising the step of repairing a beginning of the ~~first~~ data string.

5. (Currently amended) The computer-implemented method of claim 4, wherein the step of repairing the beginning of the ~~first~~ data string comprises the steps of:

~~(a)~~ determining if the first byte of the ~~first~~ data string is a second byte of a double byte character, and if the first byte of the data string is a second byte of a double byte character ~~so~~, setting a value of the first byte to shift-out and changing the attribute of ~~a~~ the first byte to ~~be~~ a shift-out character;

~~(b)~~ ~~if not,~~ determining, if the first byte of the data string is not a second byte of a double byte character, whether or not ~~if~~ the first byte of the ~~first~~ data string is a first byte of a double byte character and if the first byte of the data string is a first byte of a double byte character ~~so~~, then setting the value of the second byte to shift-out and changing the attribute of the second byte to ~~be~~ a shift-out character, and removing the first byte;

~~(c)~~ ~~if not,~~ determining, if the first byte of the data string is not a first byte of a double byte character, whether or not ~~if~~ the first byte of the

~~first~~ data string is a shift-in character, and if the first byte of the data string is a shift-in character ~~so~~, then removing the first byte of the ~~first~~ data string;

~~(d)~~ ~~if not~~ determining, if the first byte of the data string is not a shift-in character, whether or not ~~if~~ the first byte of the ~~first~~ data string is a shift-out character and if the first byte of the data string is a shift-out character ~~so~~, determining if ~~a~~ the second byte is a shift-in character, and if so, removing the first two bytes of the ~~first~~ data string.

6. (Currently amended) The computer-implemented method of claim 1, further comprising the steps of:

~~(a)~~ determining if the ~~first~~ data string is less than the fixed-length;

~~(b)~~ ~~if so, then~~ appending spaces to the end of the ~~first~~ data string so the ~~first~~ data string is left-aligned if it is determined that the data string is less than the fixed-length.

7. (Currently amended) The computer-implemented method of claim 1, further comprising the steps of:

~~(a)~~ ~~determine~~ determining if the ~~first~~ data string is less than the fixed-length;

~~(b)~~ ~~if so, then~~ prepending, if it is determined that the data string is less than the fixed length, spaces to the beginning of the ~~first~~ data string so the ~~first~~ data string is right-aligned.

8. (Currently amended) The computer-implemented method of claim 1, further comprising:

~~(a)~~ expanding the ~~first~~ data string for editing.

9. Canceled.

10. (Currently amended) The computer-implemented method of claim 8, wherein the step of expanding the ~~first~~ data string comprises the steps of:

~~(a)~~ making a copy of the ~~first~~ data string;

~~(b)~~ inserting a space for each byte that has an attribute of shift-out~~, insert a space~~;

~~(c)~~ inserting a space for each byte that has an attribute of shift-in~~, insert a space~~;

~~(d)~~ ~~represent~~ representing each single byte character as a Unicode equivalent;

~~(e)~~ ~~represent~~ representing each double byte character as a Unicode equivalent; and

~~(f)~~ ~~construct~~ constructing a byte array with the above substitutions.

11. (Currently amended) The computer-implemented method of claim 8, wherein said step of expanding the ~~first~~ data array comprises the steps of:

~~(a)~~ making a copy of the ~~first~~ data string;

~~(b)~~ inserting a parser-recognized shift-out character for each byte that has an attribute of shift-out~~, insert a parser recognized shift out character~~;

~~(c)~~ inserting a parser-recognized shift-in character for each byte that has an attribute of shift-in~~, insert a parser recognized shift in character~~;

~~(d)~~ ~~represent~~ representing each single byte character as its respective Unicode equivalent;

~~(e)~~ ~~represent~~ representing each double byte character as its respective second Unicode equivalent and a copy of the respective second Unicode equivalent; and

~~(f)~~ ~~construct~~ constructing a byte array with the above substitutions.

12. (Currently amended) The computer-implemented method of claim 8 ~~9~~, further comprising the steps of:

~~(a)~~ returning the edited ~~first~~ data string; and

~~(b)~~ recreating a Unicode string from the edited ~~first~~ data string.

13. (Currently amended) The computer-implemented method of claim 8 ~~9~~, further comprising the steps of:

~~(a)~~ returning the edited ~~first~~ data string; and

~~(b)~~ recreating a byte array of fixed-format in EBCDIC.

14. (Currently amended) A computer-implemented method of editing data having a fixed-length ~~code~~ format on a computer system that uses data having a non-fixed-length format such that after being edited, the data continues to be of the fixed-length format, the method comprising the steps of:

~~(a)~~ receiving a ~~first~~ data byte array, the data byte array for editing a fixed-length statement of the fixed-length format and being of the non-fixed-length format;

~~(b)~~ determining ~~the~~ an encoding of the ~~data byte array~~ fixed-length statement by determining a number of bytes in each of a plurality of fixed-length fields that comprise ~~a~~ the fixed-length statement;

~~(c)~~ determining a number of bytes in the fixed-length statement;

~~(d)~~ creating a ~~first~~ data string from the ~~first~~ data byte array, the data string being encoded using the determined encoding, given a starting byte position and the number of bytes in the fixed-length statement;

~~(e)~~ assigning an attribute to each byte of the ~~first~~ data string;

~~(f)~~ creating a plurality of subsets of the ~~first~~ data string; each of the subsets corresponding to a fixed-length field,

~~(g)~~ repairing ~~a~~ an end of each of the plurality of subsets if necessary by ~~first~~ (i) determining if the last byte of the subset is a second byte of

a double byte character, and if the last byte of the subset is a second byte of a double byte character ~~so~~, setting a value of the next to the last byte to shift-out and changing the attribute of ~~a~~ the second to the last byte of the subset to be a shift-out character and removing the last byte; (ii) if the last byte of the subset is not a second byte of a double byte character, then determining if the last byte of the subset is a first byte of a double byte character and if the last byte of the subset is a first byte of a double byte character ~~so~~, then setting the value of the last byte to shift-out and changing the attribute of the last byte to ~~be~~ a shift-out character; (iii) if the last byte of the subset is not a first byte of a double byte character, determining if the last byte of the subset is a shift-out character, and if the last byte of the subset is a shift-out character ~~so~~, then removing the last byte of the subset; (iv) if the last byte of the subset is not a shift-out character determining if the last byte of the subset is a shift-in character and if the last byte of the subset is a shift-in character ~~so~~, determining if ~~a second last~~ the first byte is a shift-out character, and if the first byte is a shift-out character ~~so~~, removing the last two bytes of the subset;

~~(h)~~ repairing a beginning of the each of the subsets if necessary by: (i) determining if the first byte of the subset is a second byte of a double byte character, and if the first byte of the subset is a second byte of a double byte character ~~so~~, setting a value of the first byte to shift-out and changing the attribute of ~~a~~ the first byte to ~~be~~ a shift-out character; (ii) if the first byte of the subset is not a second byte of a double byte character, determining if the first byte of the subset is a first byte of a double byte character and if the first byte of the subset is a first byte of a double byte character ~~so~~, then setting the value and changing the attribute of the second byte to ~~be~~ a shift-out

character, and removing the first byte; (iii) if the first byte of the subset is not a first byte of a double byte character, determining if the first byte of the subset is a shift-in character, and if the first byte of the subset is a shift-in character ~~so~~, then removing the first byte of the subset; (iv) if the first byte of the subset is not a shift-in character determining if the first byte of the subset is a shift-out character and if the first byte of the subset is a shift-out character ~~so~~, determining if ~~e~~ the second byte is a shift-in character, and if the second byte is a shift-in character ~~so~~, removing the first two bytes of the subset; (~~i~~ v) determining if the subset is less than the fixed-length, and if the subset is less than the fixed-length ~~so~~, determining if the subset is to be left-aligned or right-aligned;

~~(j)~~ ~~if the subset is to be left-aligned,~~ appending spaces to the end of the subset if the subset is to be left-aligned;

~~(k)~~ ~~if the subset is to be right aligned,~~ prepending spaces to the beginning of the subset if the subset is to be right-aligned; ~~and~~

~~(l)~~ combining the subsets into a second data string, the second data string being of the fixed-length format; ~~and~~

~~(m)~~ expanding the second data string for editing; and

editing the fixed-length statement using the second data string.

15 – 25 Canceled.

26. (Currently amended) A first computer system for transferring ~~the transfer of~~ data to a second computer system, the first computer system using a

non-fixed-length data format and the second computer system using a fixed-length data format, the first computer system comprising:

~~(a)~~ application means to read an original string of data not having the ~~a~~ fixed-length format;

~~(b)~~ means to input a coding specification having the ~~a~~ fixed-length format;

~~(c)~~ means, using the coding specification, to create a substring of the original string of data having a fixed-length in accordance with the fixed-length format;

~~(d)~~ means to truncate the substring;

~~(e)~~ means to repair the beginning and/or the end of the truncated substring if necessary;

~~(f)~~ means to right-align or left-align the repaired truncated substring;

~~(g)~~ means to expand the substring if the substring is shorter than the fixed-length; ~~and~~

~~(h)~~ means to edit the substring;

~~(i)~~ means to convert the edited substring to Unicode ~~receive in a second computer~~; and

~~(l)~~ means to transfer the converted, edited substring to the second computer system ~~read and decode said and adaptively reconstruct said data based~~.

27. (Currently amended) The first computer system of claim 26, further comprising:

~~(a)~~ means to convert the edited substring to a data code format having a fixed-length format.

28. (Currently amended) The first computer system of claim 26, further comprising:

~~(a)~~ means to convert the edited substring to a data code format not having a fixed-length format.

29. Canceled.

30. (New) A computer program product on a computer readable medium for editing data having a fixed format on a computer system that uses data having a non-fixed format such that after being edited, the data continues to have the fixed format, the computer program product comprising:

code means for receiving, by the computer system, a data byte array, the data byte array having the non-fixed format and including data for editing a fixed-length statement;

code means for determining an encoding of the fixed-length statement by determining a number of bytes in each of a plurality of fixed-length fields that comprise the fixed-length statement;

code means for determining a number of bytes in the fixed-length statement;

code means for creating a data string from the data byte array, the data string being encoded using the determined encoding, given a starting byte position and the number of bytes in the fixed-length statement;

code means for assigning an attribute to each byte of the data string; and

code means for editing the fixed-length statement using the data string.

31. (New) The computer program product of claim 30, further comprising code means for repairing an ending of the data string.

32. (New) The computer program product of claim 31, wherein the code means for repairing the end of the data string further comprises:

code means for determining whether or not the last byte of the data string is a second byte of a double byte character, and if so, code means for setting a value of a byte to shift-out, the byte being the second to the last byte of the double byte character and changing the attribute of the byte to a shift-out character and removing the last byte;

code means for determining, if the last byte is not a second byte of a double byte character, whether or not the last byte of the data string is a first byte of a double byte character and if so, code means for setting the value of the last byte to a shift-out character and for changing the attribute of the last byte to a shift-out character;

code means for determining, if the last byte of the data string is not a first byte of a double byte character, whether or not the last byte of the data string is a shift-out character, and if so, code means for removing the last byte of the data string;

code means for determining, if the last byte of the data string is not a shift-out character, whether or not the last byte of the data string is a shift-in character and if so, code means for determining if the second to the last byte is a shift-out character, and if so, code means for removing the last two bytes of the data string.

33. (New) The computer program product of claim 32, further comprising code means for repairing a beginning of the data string.

34. (New) The computer program product of claim 33, wherein the code means for repairing the beginning of the data string further comprises:

code means for determining whether or not the first byte of the data string is a second byte of a double byte character, and if so, code means for setting a value of the first byte to a shift-out character and for changing the attribute of the first byte to a shift-out character;

code means for determining, if the first byte of the data string is not a second byte of a double byte character, whether or not the first byte of the data string is a first byte of a double byte character and if so, code means for setting the value of the second byte to shift-out, for changing the attribute of the second byte to a shift-out character, and for removing the first byte;

code means for determining, if the first byte of the data string is not a first byte of a double byte character, whether or not the first byte of the data string is a shift-in character, and if so, code means for removing the first byte of the data string;

code means for determining, if the first byte of the data string is not a shift-in character, whether or not the first byte of the data string is a shift-out character and if so, code means for determining whether or not the second byte is a shift-in character, and if so, code means for removing the first two bytes of the data string.

35. (New) The computer program product of claim 30, further comprising:

code means for determining if the data string is less than the fixed-length;

code means for appending spaces to the end of the data string so the data string is left-aligned if the data string is less than the fixed-length.

36. (New) The computer program product of claim 30, further comprising:

code means for determining if the data string is less than the fixed-length;

code means for prepending, if it is determined that the data string is less than the fixed length, spaces at the beginning of the data string so the data string is right-aligned.